Journal of Information Technology Impact
Vol. 4, No. 3, pp. 151-160, 2004

The Impact of Hybrid Automata on System Modeling and Analysis

Isabella Kotini[1] George Hassapis[2]
Aristotle University of Thessaloniki
Greece

Abstract

The use of formal methods, techniques and tools may generally guarantee a systems' safe operation. Such a process includes system modeling with a mathematical model, which is an approximation of the physical system, and is used to make easier the next step of analysis. Formal analysis of hybrid systems, which involve mixed continuous-valued and discrete dynamics, is concerned with verifying whether a hybrid system satisfies a desired specification, like avoiding an unsafe region of the state space. In this paper, we examine the general impact of hybrid automata on modeling and analyzing hybrid systems and we discuss in particular some advantages that arise when exploiting them in the context of particular methods for verifying the reachability of states in common paradigms that have already being investigated with other classical methods. Finally, we use two case studies concerning different classes of hybrid automata to demonstrate the applicability and the efficiency of exploiting hybrid automata on system modeling and analysis.

Keywords: Formal methods, traffic control, automotive systems, chemical process control, industry circuits, robotics, power electronics circuits.

Introduction

The basic aim of software engineering is to implement complicated systems that operate in a reliable manner. The use of formal methods to check the correctness of software designs, such as mathematically-based languages, techniques and tools may generally guarantee systems' safe operation. Such a process includes modeling of the system by using a mathematical model, which is an approximation of the physical system, and is used to make easier the next step of analysis. However, the model of a system depends on its particular application, which is depicted by a number of desired or undesired states that finally constitute the set of desired system specifications.

The necessity of controlling the operation of systems on the basis of algorithms provided by computers has lead the adoption of hybrid systems. Hybrid systems involve mixed continuous-valued dynamics (i.e. the process or machine state variables) and discrete dynamics (i.e. the program state). The dynamic behavior of a hybrid system can be described with mathematical descriptions of physical phenomena, as well as laws of physics, chemistry economics, etc. A

mathematical model is important to accurately describe the phenomena of interest. In some cases, a given mathematical description must be simplified before it can be used in consequent phases of analysis and design processes. Formal analysis of hybrid systems is concerned with verifying whether a hybrid system satisfies a desired specification, like avoiding an unsafe region of the state space; the process of formal design consists of synthesizing controllers for hybrid systems in order to meet a given specification (Alur et. al, 2000). Modeling, analyzing and designing of these systems is a rather complicated process and involves the realization of control functions and the communication of embedded computers with other machines and humans.

The designer of embedded controllers wishes to make sure that certain critical specifications of the operation of a process or machine under the automatic control or guidance of an embedded computer will be satisfied. Such critical specifications may include complex descriptions of state trajectories, which are the steps that a process follows under given initial conditions. To find out whether these specifications are satisfied, graphical and mathematical models can be used in order to guarantee the correctness of the analysis. The importance of designing such systems can clearly be shown by examples of real time systems such as chemical process control, air traffic control, automotive control, robotics and automated manufacturing. However, due to the great complexity that characterizes such systems, the likelihood of designing errors becomes much greater. Furthermore, the possibility of operation errors can provoke catastrophically loss of money, time or even human life (Clarke and Wing, 1996).

Modern engineering systems can be described by the use of hybrid system models, where a computer process is used to control and coordinate several physical processes over a computer network (Koutsoukos et. al., 2000). Moreover, hybrid system models can be used as mathematical models for many important applications, such as automated highway systems (Horowitz & Varaiya, 2000; Lygeros et al., 1998) air traffic management systems (Glover and Lygeros 2004; Livadas et al., 2000; Tomlin et.al., 1998) embedded automotive controllers (Balluchi et al., 2000), manufacturing systems (Cassandras & Pepyne, 1997), chemical processes (Engell et al., 2000; Hassapis et al., 1998), robotics (Alur, et al., 2000), power electronics circuits (Senesky et. al., 2003), industry circuits (Kotini & Hassapis, 2003), etc.

In this paper, we examine the impact of hybrid automata on modeling and analyzing hybrid systems and we discuss some advantages that arise when exploiting them in conjunction with methods FSR (Finding States of Rectangular Automata**)** and FSL (Finding States of Linear Automata**)** (Kotini, Hassapis, 2003; Kotini et al., 2003) for verifying the reachability of states in common paradigms that have already being investigated with other methods, such as typical reachability analysis (Alur et. al., 1993). A main characteristic of verification methods FSR and FSL is related to the number of computations that can be predicted when a hybrid automata model is available. The aim of this approach is to find whether the states that are defined by a specification, which is expressed in propositional logic, can be reached from initial states of the hybrid automata model of a hybrid system. In contrast to other algorithms and methods, methods FSR and FSL provide an efficient way for verifying whether a hybrid system satisfies a desired specification without constructing the reachability tree of the automaton.

Related Work

In past modeling paradigms of hybrid systems, continuous dynamics were studied via differential or difference equations, and separately from discrete dynamics that were used to examined via automata, Petri nets models, logic expressions etc, resulting so to incompletely understandings of hybrid system behaviors (Antsaklis, 2000).

Recently, a great amount of research work has been published in the area of hybrid systems, concerning the study of their behavior as the result of the interactions of their discrete and continuous components. This research addresses the problem of modeling the hybrid systems at different levels of abstraction and examining certain properties of the system by searching the state space of the system for the occurrence of states that can be related with the system properties.

Different approaches have been proposed for modeling the hybrid system behavior. Early approaches tended to focus on discrete event modeling formalisms, such as finite state machines, process algebras, Petri-nets, temporal logic, etc (Branicky et al., 1998) and switched systems (Witsenhausen, 1996). Later, the approach of extracting an equivalent discrete event model of the continuous subsystem, which could be supervised by a discrete-event supervisor was proposed, (Livadas et.al., 2000), and the use of differential Petri-nets was introduced (Demongogin et al., 1998). A framework with significant potential for practical usage was the extension of the symbolic model checking approach (SMC) for finite state machines applied to hybrid systems (Macmillan, 1993); the result was the hybrid automaton (Alur et al., 1993b). Hybrid automata are generalised finite state machines, which can be used to model and analyse the behaviour of systems involving mixed continuous and discrete evolutions of the variables (Alur et al., 1993a, 1993b).

A Brief Overview of Hybrid Automata

A hybrid automaton (Henzinger, 1996; Lemmon et al., 1999) is defined by the tuple (*N, Δ, L, I, R, T*). *N* is a marked directed graph that is characterized by an ordered pair (*V, A*), where *V* is a set of vertices and *A* is a set of directed arcs between the vertices. Vertices can describe discrete states and arcs transitions between vertices. *Δ* is a set of elements that represent the continuous-time state trajectories that the hybrid system generates, as long as it stays in a discrete state, and is described by differential inclusion of the form of $\dot{x}(t) \in F(i, X)$, where *X* denotes the real valued variables and *i* is the discrete state. The elements of the set *L* are logical propositions, which can be generated recursively by the conjunction ($\wedge$), disjunction ($\vee$) and negation ($\neg$) of atomic relationships. They label the arcs and determine the conditions under which a transition can occur. Each subset, which is assigned to an arc, is called *guard.* The elements of the set *I* are relationships that label the vertices and have the form of $a \le x \le b$ or $x \in [a, b]$. They impose constraints on the continuous-time state and they are usually called *invariants*. The elements of the set *R* label also the arcs and define reset conditions to the continuous-time state and are called *resets.* The elements of the set *T* are subsets of the initial conditions associated with each vertex and called *inits.*

Based on the type of the continuous dynamics and according to their mathematical representations, there are several types of hybrid automata, such as timed, rectangular, linear and non-linear ones. A rectangular automaton has the advantage of analyzing better a hybrid system and is defined as the automaton where the differential inclusion at any i^{th} vertex has the form of $\dot{x} = [\, k_{min}, k_{max} \,]$, which is the first derivative of a variable, is given as a range of possible values, and this range does not change (Henzinger and Kopke, 1999). Therefore, in a rectangular automaton the derivative of each variable stays between two fixed bounds, which may be different in different vertices. In a linear hybrid automaton the differential inclusion at any i^{th} vertex has the form $\dot{x} = k$, where k is constant; in timed automata k is equal to 1 (all variables are clocks); in non-linear automata the inclusion is a non-linear differential.

Verifying Specifications with Hybrid Automata

The hybrid automaton seems to be an appropriate framework for modeling a hybrid system behavior (Stiver et al., 1996). The problem of verifying with an automaton model the satisfaction of specifications of a hybrid system by using the SMC method can be expressed in a more formal way as follows (Alur et al., 2000): "Given a class of hybrid automata model and a class of specification properties , the aim is to find a computational procedure that will decide if there is a state in the state space of the model that satisfies the specification and whether this state can be reached from an initial state in a finite number of computational steps."

By far the best-known computational method is the so called reachability analysis (Alur et al., 1993b). However, because of the unaccountability of the state space of a hybrid automata model, this computational method is undecidable for the general case of a hybrid automaton. Instead, the maximal class of hybrid automata with a decidable reachable states computation has been proved (Henzinger & Kopke, 1999) to be the class of rectangular automata. This finding is of significance because hybrid systems with very general dynamics can be locally approximated by using rectangular automata (Henzinger & Majumdar, 2000). Although the decidability proof makes certain that there are a finite number of computational steps, if is requested a large amount of computations, the method becomes intractable.

For this reason, in this work another approach is presented for finding whether an automaton modeling a hybrid system enters or not in a region of states that a specification requires, when it starts from a initial state, without constructing the reachabilityh tree of the automaton but examining whether a desired region of states can be reached in a finite number of automaton iterations. Automaton iteration is defined to be the sequence of discrete state changes that an automaton model takes until it comes back to the initial state. As there is always a countable duration of time that the automaton stays in a vertex or discrete state, which is determined by the invariant of the vertex and the guard of the arc emanating from this vertex, can help to derive the number of iterations of the particular automaton

Based on this observation, already proposed methods (FSR and FSL) (Kotini & Hassapis, 2003; Kotini et al., 2003) has been developed which finds algorithmically whether a reachable region of states of a hybrid automaton exists, without constructing the reachability tree of the automaton. The relevant algorithmic procedure finds the number of automaton iterations that are

needed to reach a specified region of states by utilizing derived mathematical relationships, which express the range of values that each variable takes in each vertex, as a function of the vertex duration. The vertex duration is defined as the time that the automaton stays in each vertex and it is approximated by a polynomial function of the number of the automaton iterations using curve fitting techniques. The required data for applying the curve fitting technique are obtained by running the automaton an initial number of iterations and recording the observed times.

Case studies

In order to demonstrate the applicability of methods FSR and FSL, as well as the advantages that offer on verification problems, we present two applications paradigms; the first one concerns the model of movement of a robot, which belongs to the class of rectangular automata, and the second is based on the model of a bouncing ball, which is a switched system that presents discontinuous jumps and belongs to the class of linear automata.

The Robot Rectangular Automaton

The rectangular automaton in Figure 1 models the movement of a robot in the (x, y)-plane (Henzinger and Ho, 1995). The robot can be in one of two modes: heading roughly northeast (vertex 1) or heading roughly southeast (vertex 2). In mode 1, the derivatives of variables x and y vary within the closed interval [1, 2]. In mode 2, the derivative of variable x varies between 1 and 2, while the derivative of variable y changes its sign and varies between -1 and -2. The robot changes its mode every 1 to 2 minutes, according to its clock z. Moreover, there is a wall at the x = 0 line, causing the system invariant become $x \geq 0$. The robot starts its operation from the initial state S = (Vertex 1, x = y = z = 0).

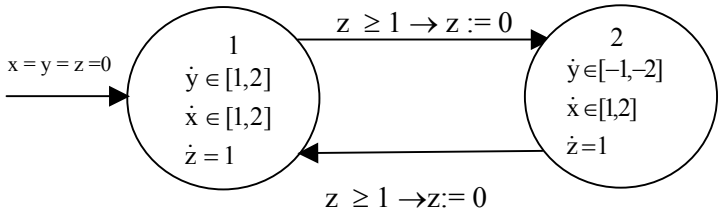

Figure 1. The rectangular automaton of a robot

The aim of verifying the specifications is to check whether the robot can reach, after starting from the initial state S, the final position T = ($x = 9 \wedge y = 12$). As it is stated in (Henzinger & Ho 1995), the application of the reachability analysis method (forward reachability analysis) does not terminate. In contrast, the results that the method FSR gives are presented in Table 1.

The FSR method (Kotini and Hassapis, 2003) answers to the problem addressed before, giving a result with few computations. For x = 9 and y = 12 there is no positive integer number *n* satisfying the inequalities given in Table 1 for each vertex, which means that the automaton will never reach the unsafe region expressed by the final region T.

Table 1. Values of variables x and y in vertices 1 and 2

Vertex 1	Vertex 2
$2n - 1 \leq x'_{1,n} \leq 4n - 2$	$2n \leq x'_{2,n} \leq 4n$
$2 - n \leq y'_{1,n} \leq n + 1$	$n \leq y'_{2,n} \leq -n$

The Bouncing Ball Linear Automaton

The bouncing ball is a switched system in which the continuous state makes discontinuous jumps on the switching boundary (Lemmon et. al. 1999). Due to an elastic collision the ball's velocity vector makes an instantaneous sign change upon hitting the floor. The automaton that models this behavior has three variables x, y and z and two vertices: vertex 1 (Down) and vertex 2 (Up). Variable x represents the horizontal distance of the ball from a reference point, variable y represents the distance of the ball from the floor, and variable z represents the "kinetic energy" of the ball. Suppose that the ball is dropped from a position with co-ordinates $x = 14 \wedge y = 4$ to the direction of the reference point; that is, the initial conditions denoted are $x = 14$, $y = 4$ and $z = 0$. While the ball is falling (vertex 1), its kinetic energy is increasing ($\dot{z} = 1$); when the ball hits the floor ($y = 0$), it loses half of its energy and bounces back up (vertex 2); on the way up, the kinetic energy decreases until it becomes 0 and the ball starts to fall again. The linear automaton, which describes bouncing-ball (Henzinger & Ho, 1995), is shown in the following Figure 2.

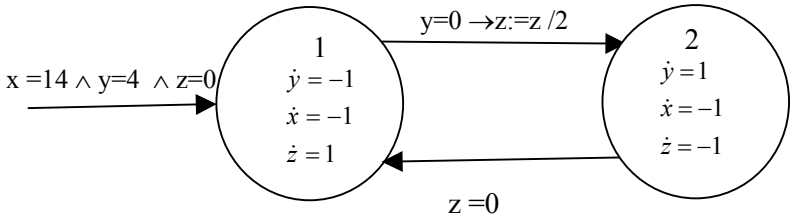

Figure 2. The bouncing-ball automaton.

The aim is to prove that the ball never reaches the region that is defined by the co-ordinates T = (Vertex 1, $x \leq 2 \wedge y = 0$) of the floor. As it is stated in (Henzinger & Ho 1995), the application of the reachability analysis method (forward and backward reachability analysis) does not terminate. However, the results of applying the method FSL (Kotini et al., 2003), which particularly regards the duration in each vertex, are shown in Table 2.

Table 2. Verification of linear automaton.

Vertex 1	Vertex 2
$\delta_{1,n} = \frac{4}{2^n}$	$\delta_{2,n} = \frac{2}{2^n}$

Applying the curve fitting method on the durations of the vertices for a number of 1000 samples, we found the values of variables x and y of vertex 1, which correspond to an estimation error of 6%. A graphical representation of the resulting values is depicted in Figures 3 and 4, for variables x and y, respectively.

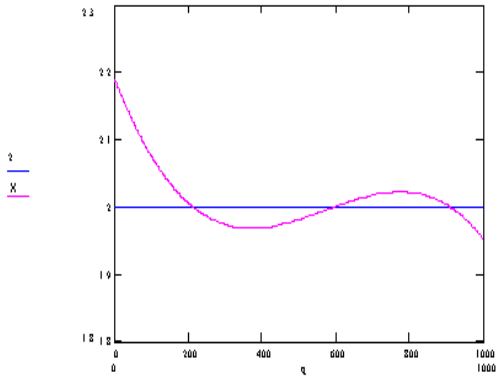

Figure 3. Variable x in vertex 1

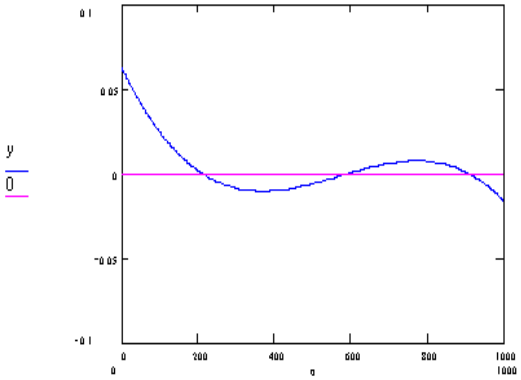

Figure 4. Variable y in vertex 1

The polynomial fuctions of values x and y in Vertex 1 are:

$$x_{1,n} = 14 - 7.874 - 9.438*10^{-4}*n + 1.886*10^{-6}*n^2 - 1.1*10^{-9}*n^3 - 3.937 - 4.719*10^{-4}*(n-1) + 9.431*10^{+7}*(n-1)^2 - 5.5*10^{-10}*(n-1)^3$$

$$y_{1,n} = 4 - 7.874 - 9.438*10^{-4}*n + 1.886*10^{-6}*n^2 - 1.1*10^{-9}*n^3 + 3.937 + 4.719*10^{-4}*(n-1) - 9.431*10^{+7}*(n-1)^2 + 5.5*10^{-10}*(n-1)^3$$

Since in vertex 1, the variable x cannot be less than 2 and y equal to zero, for any positive integer value of *n* (where n is the number of iterations), one can conclude that the region T can not be reached.

Conclusions

In this paper, we illustrated the use of hybrid automata to efficiently model and verify the specifications of hybrid systems. Furthermore, we argued that exploiting hybrid automata with methods FSR and FSL can give answers in undecidable classes of well-known verification problems, resulting so in significant impacts on systems modeling and analysis.

The core idea of these methods is based on algebraic mathematical relationships for finding whether a region of states that can be associated with the specifications of linear and rectangular automata models exists, without the need of constructing the reachability tree of the automaton. These relationships express the range of values that each variable takes in each vertex as a function of the vertex duration. This duration in turn is approximated by a polynomial function of the number of the automaton iterations using curve-fitting techniques. The vertex duration is defined as the time that the automaton stays in each vertex. Two case studies of different classes of hybrid automata concerning, the movement of a robot (for the class of rectangular automata) and the bouncing ball automaton (for the class of linear automata) have been used to demonstrate the applicability and the efficiency of using hybrid automata, in particular with methods FSR and FSL, on system modeling and analysis.

Hybrid systems modeling and analysis is a technical field that exploits various methods and concepts from computer science and traditional system science. It is still in an early stage of development but there are already important results that reveal their positive impact on proving the safe operation of hybrid systems on various areas, including traffic control, automotive systems, chemical process control, industry circuits, robotics, and power electronics circuits.

References

Alur, R., Courcoubetis, C., & Dill, D. (1993a). Model checking in dense real time systems, *Information and Computation*, 104, 2-34.

Alur, R., Courcoubetis, C., Henzinger, T. A., Ho, P-H. (1993b). Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems, in *Proc. Hybrid Systems I,* Lecture Notes in Computer Science, 736, 209-229.

Alur, R., Grosu, R., Hur, Y., Kumar, V., & Lee, I. (2000). Modular specification of hybrid systems in CHARON. In *hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, 1790. Springer-Verlag.

Antsaklis, P. (2000). A Brief Introduction to the Theory and Applications of Hybrid systems. In *Proceedings of the IEEE, Special Issue on Hybrid Systems: Theory and Applications*, 88(7), 879-887.

Ashish, T., & Gauran, K. (2002). Series of abstractions for Hybrid Automata, *Hybrid Systems: Computation and Control CHSCC,* 25-27.

Balluchi A., Benvenuti, L., DiBenedetto, M., Pinello, C., & Sangiovanni-Vincentelli, A. (2000). Automotive engine control and hybris sytems: Challenges and opportunities. In *Proceedings of the IEEE*, *Special Issue on Hybrid Systems: Theory and Applications*, 88(7), 888-912.

Branicky, V., Borkar, S., Mitter, S. K. (1998). A unified framework for hybrid control, *IEEE Transactions On Automatic Control*, 43(1), 31-45.

Clarke, E. M. & Wing, J. M. (1996). Formal Methods: State of the Art and Future Directions. *In ACM Computing Surveys,*28(4), 626-643.

Demongogin, I., Koussoulas, N.T. (1998). Differential Petri Nets: Representing Continuous Systems in a Discrete-Event World, *IEEE Trans.Autom.Control*, 43(4), 573-579.

Engell, S., Kowalewski, S., Schulz, C., & Stursberg, O. (2000). Continous-discrete interactions in chemical processing plants. *In Proceedings IEEE*, , *Special Issue on Hybrid Systems: Theory and Applications,* 88, 1080-1068.

Glover, W., & Lygeros, J. (2004). A Stochastic Hybrid Model for Air Traffic Control Simulation. In *hybrid Systems: Computation and Control*, LNCS 2993, 372-386.

Hassapis, G., Kotini, I., & Doulgeri, Z. (1998). Validation of a SFC Software Specification by Using Hybrid Automata. In *9th Symposium on Information Control in Manufacturing*

Henzinger, T. A., (1996). "The Theory of Hybrid Automata," in: Proc. *11th Annual IEEE Symposium on Logic in Computer Science,* 278-292. IEEE Computer Society Press.

Henzinger, T. A., & Ho, Pei-Hsin. (1995). A Note on Abstract Interpretation Strategies for Hybrid Automata. In *Proc. Hybrid Systems II, Lecture Notes in Computer Science,* 999, 252-264. Springer-Verlag.

Henzinger, T. A. Majumdar, R. (2000). Symbolic Model Checking for Rectangular Hybrid Systems, S. Graf and M. Schwartzbach (eds.):*TACAS/ETAPS*, LNCS 1785, 142-156.

Henzinger, T. A., & Kopke, P. T. (1999). Discrete-time control for rectangular hybrid automata, *Theoretical Computer Science*, 221, 369-392,

Horowitz, R., & Varaiya, P. (2000). Control design of an automated highway system. In *Proceedings IEEE*, *Special Issue on Hybrid Systems: Theory and Applications*, 88, 1026-1049.

Kotini, I., & Hassapis, G. (2003). Modelling and Performance Evaluation of Hybrid Systems. In *Proceedings of 1st South-East European Workshop on Formal Methods* (SEEFM'03), Greece, 21-35.

Kotini, I., Hassapis, G. & Mavridis, I. (2003). "Verifying Specifications in Hybrid Automata Models of Systems". In *Proceedings of 7rd WSEAS International Multiconference on Circuits, Systems, Communications and Computers* (CSCC 2003), Greece, 146-151.

Kotini, I., Hassapis, G., & Mavridis, I. (2003). Verification of Hybrid System Specifications using Linear Hybrid Automata. In *Proceedings of 7rd WSEAS International Multiconference on Circuits, Systems, Communications and Computers*, Corfu, Greece, 112-118.

Koutsoukos, X., Antsaklis, P., Stiver J., & Lemmon, M. (2000). Supervisory Control of Hybrid Systems. In *Proceedings of the IEEE*, *Special Issue on Hybrid Systems: Theory and Applications* 88(7), 1026-1049.

Lemmon, M. D., Xe, K. X., & Markovsky, I. (1999). Supervisory Hybrid Systems, *IEEE Control Systems*, 19, 42-55,

Livadas, C., Lygeros, J., & Lynch, N. (2000). High-level modeling and analysis of the traffic alert and collision avoidance systems. *In Proceedings IEEE*, *Special Issue on Hybrid Systems: Theory and Applications*, 88(7).

Lygeros, J., Pappas, G. J., & Sastry, S. (1998). An approach to the verification of the Center-TRACON Automation System. In *Hybrid systems: Computation and Control*, 1386, 289-304.

Lygeros J., Johansson, K. H., Simic, S.N., Zhang, J., Sastry, S. (2003). Dynamical Propetries of Hybrid Automata. In *IEEE Trans. Automat. Control ,* 48(1).

Macmillan, K. (1993). *Symbolic Model Checking*, Kluwer Academic,

Senesky M., Eirea, G., Koo, J. (2003). Hybrid Modelling and Control of Power Electronics. In *Hybrid Systems: Computation and Control,* LNCS 2623, 450-465.

Stiver, J. A., Antsaklis, P. J., & Lemmon, M. D. (1996). A logical DES approach to the design of hybrid control systems, *Mathematical Computer Modeling*, 23, 55-76.

Tomlin C., Pappas G., & Sastry S. (1998). Conflict resolution for air traffic management: A study in multiagent hybrid systems. In *IEEE Transactions of Automatic Control,* 43(4).

Witsenhausen, H. S. (1966). A class of hybrid -state continuous time dynamic systems, *IEEE Transactions of. Automatic Control*. 11(2), 161-167

---

[1] Dr. Isabella Kotini is a secondary school teacher and teaches computer science at the Aristotle University of Thessaloniki. She can be reached at Kassandrou 141, 54634, Thessaloniki, Greece. Email: ikotin@uom.gr; Phone: +(30) 2310-204426

[2] Dr. George Hassapis is a Professor at the Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Greece. He can be reached at Department of Electrical and Computer Engineering, Aristotle University**,** 54124, Thessaloniki, Greece. Email: chasapis@eng.auth.gr; Phone: +(30) 2310-996324